ISSN : 2407 - 3911

PENGUJIAN APLIKASI MENGGUNAKAN *BLACK BOX TESTING BOUNDARY VALUE ANALYSIS* (Studi Kasus : Aplikasi Prediksi Kelulusan SNMPTN)

M. Sidi Mustaqbal[1], Roeri Fajri Firdaus[2], Hendra Rahmadi[3]
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Widyatama
Jl. Cikutra no. 204 A,Bandung, 40125
Email : sidi.mustaqbal@gmail.com[1], rff.firdaus@gmail.com[2], eenk.hendra@yahoo.co.id[3]

Abstrak

Pengujian adalah suatu proses pelaksanaan suatu program dengan tujuan menemukan suatu kesalahan. Suatu kasus test yang baik adalah apabila test tersebut mempunyai kemungkinan menemukan sebuah kesalahan yang tidak terungkap. Suatu test yang sukses adalah bila test tersebut membongkar suatu kesalahan yang awalnya tidak ditemukan. Salah satu dari jenis pengujian yang ada adalah *Black Box Testing*.

Pada penelitian ini dicoba diterapkan pengujian dengan menggunakan teknik *Black Box Testing*. Metoda *Black Box Testing* terdiri atas beberapa cara antara lain *Equivalence Partitioning, Boundary Value Analysis, Comparison Testing, Sample Testing, Robustness Tesing*, dan lain-lain. Diantara sekian banyak cara pengujian tersebut, pada penelitian ini dipilih cara pengujian *Boundary Value Analysis*.

Boundary Value Analysis merupakan cara pengujian dengan menentukan nilai batas bawah dan batas atas dari data yang ingin diuji. Pengujian ini dilakukan pada fungsi tambah kelas pada Aplikasi Prediksi Kelulusan SNMPTN.

Hasil pengujian menunjukkan bahwa masih terdapat banyak kekurangan saat melakukan validasi data yang akan dimasukkan, sehingga dapat menyebabkan data yang disimpan pada *database* tidak sesuai dengan data yang diharapkan. Hasil pengujian dapat dijadikan masukan untuk memperbaiki aplikasi.

Kata kunci :

Pengujian, *Blackbox Testing*, *Boundary Value Analysis*.

Abstract

Testing is a process of implementing a program with the aim of finding faults. A good test case is that if the test is able to find an error that was not revealed. A successful test is when that test can find the error that was not found at the beginning. One of the types of testing is called Black Box Testing.

The aim of this research is trying to apply the blackbox testing on a software. Black Box Testing provide us with some techniques to run the test, such as: Equivalence Partitioning, Boundary Value Analysis, Comparison Testing, Sample Testing, Robustness Tesing, and others. Among the many ways of testing, we choose the software testing using Boundary Value Analysis techniques.

Boundary Value Analysis is a way of testing by determine the value of the lower limit and upper limit of the data that want to test. Testing was conducted on the "Add Class" functionality of the developed software called "Aplikasi Prediksi Kelulusan SNMPTN".

The test results show that there are still many shortcomings when performing data validation, so it can cause unexpected data is saved into database. The result can give the feedback for developer to improve the application features.

Keywords:

Testing, Testing Blackbox, Boundary Value Analysis.

I. PENDAHULUAN

Aplikasi Prediksi Kelulusan Seleksi Nasional Masuk Perguruan Tinggi Negeri (SNMPTN) merupakan aplikasi yang dibuat untuk membantu para guru bimbingan konseling ketika membimbing siswa/siswi nya untuk memilih perguruan tinggi mana yang diperkirakan pas dengan prestasi mereka selama ini. Dengan aplikasi ini, diharapkan tingkat kelulusan siswa pada SNMPTN dapat meningkat.

Pada aplikasi ini terdapat beberapa modul, antara lain modul pengelolaan kelas, pengelolaan siswa, pengelolaan data nilai siswa, pengelolaan data Perguruan Tinggi Negeri dan Jurusannya, Pengurutan peringkat siswa berdasarkan nilai rapot, dan lain-lain.

Setelah aplikasi selesai dibuat, maka perlu dilakukan pengujian untuk memastikan semua proses berjalan sesuai dengan yang diinginkan. Pengujian adalah suatu proses pelaksanaan suatu program dengan tujuan menemukan suatu kesalahan. Suatu kasus test yang baik adalah apabila test tersebut mempunyai kemungkinan menemukan sebuah kesalahan yang tidak terungkap. Suatu test yang sukses adalah bila test tersebut membongkar suatu kesalahan yang awalnya tidak ditemukan. Tujuan utama dari pengujian adalah untuk mendesain test yang secara sistematik membongkar jenis kesalahan dengan usaha dan waktu minimum.

II. KAJIAN LITERATUR

II.1 Pengujian *Software*

Pengujian *software* sangat diperlukan untuk memastikan *software*/aplikasi yang sudah/sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Pengembang atau penguji *software* harus menyiapkan sesi khusus untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya. Pengujian atau testing sendiri merupakan elemen kritis dari jaminan kualitas perangkat lunak dan merupakan bagian yang tidak terpisah dari siklus hidup pengembangan *software* seperti halnya analisis, desain, dan pengkodean. (Shi, 2010)

Pengujian software haruslah dilakukan dalam proses rekayasa perangkat lunak atau *software engineering*. Sejumlah strategi pengujian *software* telah diusulkan dalam literatur. Semuanya menyediakan template untuk pengujian bagi pembuat *software*. Dalam hal ini, semuanya harus memiliki karakteristik umum berupa (Bhat and Quadri, 2015) :

1. Testing dimulai pada level modul dan bekerja keluar ke arah integrasi pada sistem berbasiskan komputer
2. Teknik testing yang berbeda sesuai dengan poin-poin yang berbeda pada waktunya
3. Testing diadakan oleh pembuat/pengembang *software* dan untuk proyek yang besar oleh group testing yang independent
4. *Testing* dan *Debugging* adalah aktivitas yang berbeda tetapi *debugging* harus diakomodasikan pada setiap strategi *testing*

Pengujian *software* adalah satu elemen dari sebuah topik yang lebih luas yang sering diartikan sebagai Verifikasi dan Validasi (V&V)

- Verifikasi: menunjuk kepada kumpulan aktifitas yang memastikan bahwa *software* telah mengimplementasi sebuah fungsi spesifik.
- Validasi: menunjuk kepada sebuah kumpulan berbeda dari aktivitas yang memastikan bahwa software yang telah dibangun dapat ditelusuri terhadap kebutuhan customer.

Definisi V&V meliputi banyak aktifitas SQA (*software quality assurance*), termasuk review teknis formal, kualitas dan audit konfigurasi, monitor performance. Terdapat beberapa tipe yang berbeda dalam pengujian software yang meliputi studi kelayakan dan simulasi. (Bhat and Quadri, 2015):

1. Metode *software engineering* menyediakan dasar dari mutu yang mana yang akan dipakai.
2. Metode *Analysis, design and Construction* berupa tindakan untuk meningkatkan kualitas dengan menyediakan teknik yang seragam dan hasil yang sesuai dengan keinginan.
3. Metode *Formal Technical Reviews* menolong untuk memastikan kualitas kerja produk merupakan hasil konsekuensi dari setiap langkah *software engineering*.
4. Metode *Measurement* diberlakukan pada setiap elemen dari konfigurasi software
5. Metode *Standards and Procedures* membantu untuk memastikan keseragaman dan formalitas dari SQA untuk menguatkan dasar “filosofi kualitas total”

6. Metoda *Testing* menyediakan cara terakhir dari tingkat kualitas mana yang dapat dicapai dan dengan praktis dapat mengetahui letak error.

Davids menyarankan satu set prinsip pengujian:

1. Semua test harus dapat dilacak ke kebutuhan pelanggan.
2. Test harus direncanakan dengan baik sebelum pengujian mulai.
   a. Prinsip Pareto berlaku untuk pengujian
   b. 80% dari semua kesalahan yang terungkap selama pengujian akan mudah dapat dilacak dari 20% semua modul program.
3. Pengujian seharusnya mulai "dari yang kecil" dan pengujian perkembangan ke arah "yang besar".
4. Pengujian menyeluruh adalah tidak mungkin. Paling efektif, pengujian harus diselenggarakan oleh suatu pihak ketiga mandiri.

Langkah-langkah pengujian software ada 4 yaitu:

1. *Unit testing*-testing per unit yaitu mencoba alur yang spesifik pada struktur modul kontrol untuk memastikan pelengkapan secara penuh dan pendeteksian error secara maksimum
2. *Integration testing* – testing per penggabungan unit yaitu pengalamatan dari isu-isu yang diasosiasikan dengan masalah ganda pada verifikasi dan konstruksi program
3. *High-order test* yaitu terjadi ketika software telah selesai diintegrasikan atau dibangun menjadi satu –tidak terpisah-pisah
4. *Validation test* yaitu menyediakan jaminan akhir bahwa software memenuhi semua kebutuhan fungsional, kepribadian dan performa.

Tom Gilb menyatakan bahwa prosedur yang harus digunakan jika ingin mengimplementasikan strategi testing software yang sukses (Bhat and Quadri, 2015):

1. Menetapkan seluruh kebutuhan produk *software* dalam perhitungan sebelum memulai testing
2. Status obyek testing harus jelas
3. Memahami pengguna *software* dan mengembangkan sebuah profil untuk setiap kategori user
4. Mengembangkan rencana testing yang menekankan pada "*rapid cycle testing*"
5. Membangun *software* yang sempurna yang didesain untuk menguji dirinya sendiri
6. Menggunakan tinjauan ulang yang formal sebagai filter sebelum pengujian
7. Melakukan tinjauan ulang secara formal untuk menilai strategi tes dan kasus tes itu sendiri
8. Mengembangkan pendekatan peningkatan yang berkelanjutan untuk proses testing

Ada beberapa jenis pengujian perangkat lunak, antara lain (Khan, 2011):

1. Pengujian *white box* adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Secara sekilas dapat diambil kesimpulan *white box testing* merupakan petunjuk untuk mendapatkan program yang benar secara 100%,
2. *Black-Box Testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.

II.2 White Box Testing

White Box Testing adalah salah satu cara untuk menguji suatu aplikasi atau *software* dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang di buat ada yang salah atau tidak. Kalau modul yang telah dan sudah di hasilkan berupa output yang tidak sesuai dengan yang di harapkan maka akan dikompilasi ulang dan di cek kembali kode-kode tersebut hingga sesuai dengan yang diharapkan (Nidhra and Dondetti, 2012).

Kasus yang sering menggunakan *white box testing* akan di uji dengan beberapa tahapan yaitu:

1. Pengujian seluruh keputusan yang menggunakan logikal.
2. Pengujian keseluruh loop yang ada sesuai batasan-batasannya.
3. Pengujian pada struktur data yang sifatnya internal dan yang terjamin validitasnya.

Kelebihan *White Box Test*ing antara lain (Nidhra and Dondetti, 2012) :

1. Kesalahan Logika

Menggunakan sintax ‘if’ dan sintax pengulangan. Langkah selanjutnya metode *white box testing* ini akan mencari dan mendeteksi segala kondisi yang di percaya tidak sesuai dan mencari kapan suatu proses perulangan di akhiri.

2. Ketidaksesuaian Asumsi

   Menampilkan dan memonitor beberapa asumsi yang diyakini tidak sesuai dengan yang diharapkan atau yang akan diwujudkan, untuk selanjutnya akan dianalisa kembali dan kemudian diperbaiki.

3. Kesalahan Pengetikan

   Mendeteksi dan mencaribahasa-bahasa pemograman yang di anggap bersifat *case sensitif*.

Kelemahan *White Box Testing* adalah pada perangkat lunak yang jenisnya besar, metode *white box testing* ini dianggap boros karena melibatkan banyak sumberdaya untuk melakukannya. (Nidhra and Dondetti, 2012)

II.3 *Black Box* Testing

Black Box Testing berfokus pada spesifikasi fungsional dari perangkat lunak. *Tester* dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program.

Black Box Testing bukanlah solusi alternatif dari *White Box Testing* tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *White Box Testing*.

Black Box Testing cenderung untuk menemukan hal-hal berikut:
1. Fungsi yang tidak benar atau tidak ada.
2. Kesalahan antarmuka *(interface errors).*
3. Kesalahan pada struktur data dan akses basis data.
4. Kesalahan performansi *(performance errors).*
5. Kesalahan inisialisasi dan terminasi.

Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:
1. Bagaimana fungsi-fungsi diuji agar dapat dinyatakan valid?
2. Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
3. Apakah sistem sensitif pada input-input tertentu?
4. Bagaimana sekumpulan data dapat diisolasi?
5. Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?
6. Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Saat ini terdapat banyak metoda atau teknik untuk melaksanakan Black Box Testing, antara lain:
1. *Equivalence Partitioning*
2. *Boundary Value Analysis/Limit Testing*
3. *Comparison Testing*
4. *Sample Testing*
5. *Robustness Testing*
6. *Behavior Testing*
7. *Requirement Testing*
8. *Performance Testing*
9. Uji Ketahanan *(Endurance Testing)*
10. Uji Sebab-Akibat *(Cause-Effect Relationship Testing)*

II.4 *Boundary Value Analysis / Limit Testing*

Boundary value analysis adalah salah satu teknik *black box testing* yang melakukan pengujian pada batas atas dan batas bawah nilai yang diisikan pada aplikasi. Beberapa prinsip yang mendasari pada *boundary value analysis* (BVA) yaitu :
1. Banyak kesalahan terjadi pada kesalahan masukan.
2. BVA mengijinkan untuk menyeleksi kasus uji yang menguji batasan nilai input.
3. BVA merupakan komplemen dari *equivalence partitioning.* Lebih pada memilih elemen-elemen di dalam kelas ekivalen pada bagian sisi batas dari kelas.
4. Contoh:
   a. Untuk rentang yang dibatasi a dan b maka uji (a-1), a, (a+1), (b-1), b, (b+1).
   b. Jika kondisi input mensyaratkan sejumlah n nilai maka uji dengan sejumlah (n-1), n dan (n+1) nilai.
   c. Aplikasikan dua aturan sebelumnya pada kondisi output (buat table pengujian hasil outputnya untuk nilai maksimal dan minimal).
   d. Jika struktur data internal dari program memiliki cakupan (misal: ukuran buffer, batas array) gunakan data input yang menguji batas cakupan.

Secara umum, aplikasi BVA dapat dikerjakan secara generic. Bentuk dasar implementasi BVA adalah untuk menjaga agar satu variable berada pada nilai nominal (normal atau rata-rata) dan mengijinkan variable lain diisikan dengan nilai ekstrimnya. Nilai

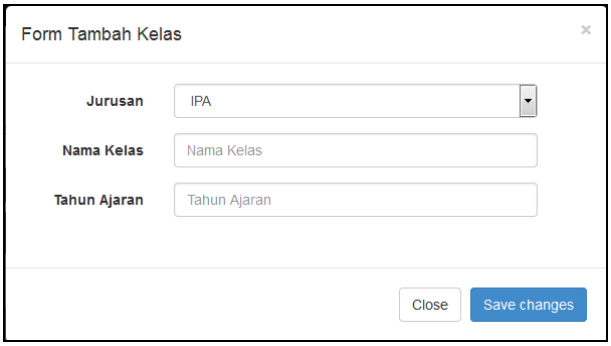
yang digunakan untuk menguji keekstriman data adalah:
Min -------- minimal
Min+ -------- di atas minimal
Nom -------- rata-rata
Max- -------- tepat di bawah maksimum
Max -------- Maksimum

Sebagai contoh, misalnya akan dientrikan data tanggal. Data tanggal memiliki tiga variable yaitu tanggal, bulan dan tahun. Maka untuk ketiga variable tersebut, dapat diambil kondisi berikut :
$1 \leq$ tanggal ≤ 31
$1 \leq$ bulan ≤ 12
$1812 \leq$ tahun ≤ 2016

Maka untuk setiap entri data di luar angka di atas akan menampilkan pesan "Tanggal yang anda isikan salah".

III. PENGUJIAN BOUNDARY VALUE ANALYSIS

Berdasarkan penjelasan sebelumnya, dicoba diterapkan teknik BVA untuk menguji aplikasi yaitu "Aplikasi Prediksi Kelulusan SNMPTN". Aplikasi ini sendiri memuat beberapa fungsi dan modul, namun pada artikel ini sebagai contoh akan dibahas hasil pengujian pada salah satu fungsionalitas yaitu "Tambah Kelas". Fungi "Tambah Kelas" terdiri atas satu panel entri data seperti pada gambar 1. Pada form ini terdapat 3 field entri data yaitu jurusan, nama kelas dan tahun ajaran.



Gambar 1. Form Tambah Kelas

Data hasil entri form di atas akan disimpan pada tabel dengan format seperti pada tabel 1.

Berdasarkan form di atas, kemudian dilakukan pengujian dengan menyiapkan beberapa data uji. Dari bentuk form di atas, contoh pengujian akan dilakukan pada dua field yaitu nama kelas dan tahun ajaran. Hal ini karena field "Jurusan" berupa dropbox yang sudah tersedia isinya. Untuk kedua field tersebut, akan dibuat scenario pengujian dan hasil ujinya seperti pada tabel 2 hingga tabel 7.

Tabel 1. Struktur Tabel "Kelas"

Nama Field	Type Data	Constraint
Id_kelas	Int(11)	PK
Nama kelas	Varchar(10)	
Tahun ajaran	Varchar(11)	
Id guru	Int(11)	FK(Guru)
Id jurusan	Int(11)	FK(Jurusan)

A. Pengujian field "Nama Kelas"

Aturan entri data A.1 : harus terdiri dari 3 kata.

Tabel 2. Hasil Uji Aturan A.1

Sample Data	Expected Result	Result	Conclusion
IPA 12	F	T	Failed
12 IPA 1	T	T	Success
12 IPS 2 A	F	T	Failed

Aturan entri data A.2 : Kata pertama merupakan angka 12

Tabel 3. Hasil Uji Aturan A.2

Sample Data	Expected Result	Result	Conclusion
11	F	T	Failed
12	T	T	Success
13	F	T	Failed

Aturan entri data A.3 : Kata kedua merupakan nama jurusan (IPA atau IPS)

Tabel 4. Hasil Uji Aturan A.3

Sample Data	Expected Result	Result	Conclusion
AAA	F	T	Failed
IPA	T	T	Success
IPS	T	T	Success
MIPA	F	T	Failed

Aturan entri data A.4: Kata ketiga hanya berupa angka dari 1 sampai 10

Tabel 5. Hasil Uji Aturan A.4

Sample Data	Expected Result	Result	Conclusion
0	F	T	Failed
1	T	T	Success
2	T	T	Success
9	T	T	Success
10	T	T	Failed
11	F	T	Failed

B. Pengujian field "Tahun ajaran"

Aturan entri data B.1: Harus berupa angka dengan panjang 4 karakter

Tabel 6. Hasil Uji Aturan B.1.

Sample Data	Expected Result	Result	Conclusion
203	F	T	Failed
2017	T	T	Success
20123	F	T	Failed

Aturan entri data B.2: Tahun ajaran harus di antara 2011 dan 2015

Tabel 6. Hasil Uji Aturan B.2.

Sample Data	Expected Result	Result	Conclusion
2010	F	T	Failed
2011	T	T	Success
2012	T	T	Success
2014	T	T	Success
2015	T	T	Success
2016	F	T	Failed

Berdasarkan hasil uji pada satu form di atas, dapat disiapkan beberapa kasus data uji. Pada contoh di atas, terdapat dua field yang akan diuji, dengan masing-masing field memuat minimal 2 aturan uji. Untuk satu aturan uji, perlu disiapkan minimal 3 data uji, sehingga total data uji yang perlu disiapkan untuk kasus di atas adalah 2 field x 2 aturan x 3 data uji yaitu 12 data uji. Hal ini dapat memberikan gambaran jumlah data uji yang harus disiapkan untuk melakukan blackbox testing dengan metode BVA. Hasil pengujian memperlihatkan bahwa aplikasi ini masih memiliki beberapa kekurangan yaitu belum lengkapnya proses validasi data sehingga masih perlu penyempurnaan dengan menambah fungsi validasi.

IV. KESIMPULAN

Setelah melakukan pengujian pada fungsionalitas tambah kelas dengan menggunakan metode pengujian *Blackbox Testing Boundary Value Analysis* dapat ditarik kesimpulan bahwa:

1. Metode *Blackbox Testing* merupakan salah satu metode yang mudah digunakan karena hanya memerlukan batas bawah dan batas atas dari data yang di harapkan,
2. Estimasi banyaknya data uji dapat dihitung melalui banyaknya field data entri yang akan diuji, aturan entri yang harus dipenuhi serta kasus batas atas dan batas bawah yang memenuhi.
3. Setelah melakukan pengujian diketahui bahwa fungsionalitas masih bisa berjalan namun masih dapat menerima masukan data yang tidak diharapkan sehingga dapat menyebabkan data yang disimpan kurang valid.
4. Berdasarkan hasil uji dengan metoda BVA maka fungsi entri data perlu dilengkapi dengan beberapa proses validasi data untuk menjamin akurasi entri data sesuai fungsional yang diinginkan.

DAFTAR PUSTAKA

Shi, Mingtao, 2010, Software Functional Testing from the Perspective of Business Practice Computer and Information Science, www.ccssenet.org/cis

Bhat, A, and Quadri, S.M.K, 2015, Equivalence Class Partitioning and Boundary Value Analysis = A review, 2nd International Conference on Computing for Sustainable Global Development (INDIACom)

Khan, Mohd Ehmer, 2011, Different Approach to Blackbox Testing Technique for Finding Error, International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.4, October 2011

Nidhra, Srinivas, and Dondeti, Jagruthi, 2012, Blackbox and Whitebox Testing Techniques - A Literature Review, International Journal of Embedded Systems and Applications (IJESA) Vol.2, No.2, June 2012